LA-UR- 09-06299

Approved for public release; distribution is unlimited.

Title:	Advances in Kinetic Plasma Simulation with VPIC and Roadrunner
Author(s):	Kevin Bowers*, Brian Albright, Lin Yin, Bill Daughton, Vadim Roytershteyn, Ben Bergen and Tom Kwan
Intended for:	21st International Conference on Numerical Simulation of Plasmas, Lisbon, Portugal

Los Alamos
NATIONAL LABORATORY
EST. 1943

ICNSP 09

Topic/Type: 1. Plasma Simulation, Invited

Advances in ~~petascale~~ kinetic plasma simulation with VPIC and Roadrunner

K. J. Bowers, B. J. Albright, L. Yin, W. Daughton, V. Roytershteyn, B. Bergen, T. J. T. Kwan

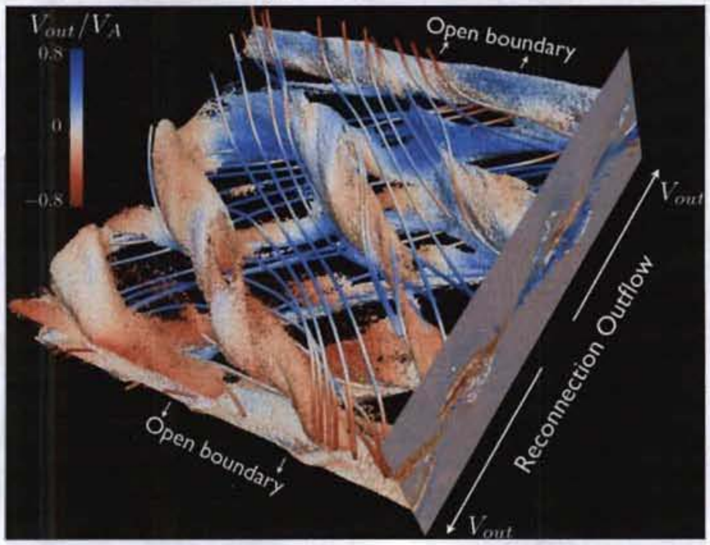
Los Alamos National Lab / D. E. Shaw Research

VPIC, a first-principles 3d electromagnetic charge-conserving relativistic kinetic particle-in-cell (PIC) code, was recently adapted to run on Los Alamos\'s Roadrunner, the first supercomputer to break a petaflop (quadrillion floating point operations per second) in the TOP500 supercomputer performance rankings. Due to physical limitations, moving data between and even within modern processors is more time consuming than performing basic computations. Typical PIC implementations require more data motion per computation than other methods often used in supercomputing (e.g. dense matrix, molecular dynamics N-body and Monte-Carlo calculations), but, unlike traditional codes, VPIC was designed from the ground up to minimize data motion. As a result, VPIC can more fully exploit the potential of petascale resources like Roadrunner. For example, VPIC can perform 0.162 billion cold particles pushed and charge-conserving accumulated per second on the heterogeneous multi-core IBM Cell eDP processors used in Roadrunner---equivalent to 0.517 petaflop (s.p.) on all of Roadrunner. During a parameter study of particle trapping physics within the laser-driven hohlraum of inertial confinement fusion experiments, we measured end-to-end sustained performance exceeding 0.374 Pflop/s (s.p.) on 122,240 processing cores (17 of Roadrunner\'s 18 connected units).

Petascale supercomputers like Roadrunner are enabling VPIC simulations of numerous plasma physics phenomena at unprecedented fidelity and scale---using trillions of particles, billions of mesh points and hundreds of thousands processing of cores. We summarize VPIC\'s modeling capabilities, VPIC\'s optimization techniques and Roadrunner\'s computational characteristics. We then discuss three applications enabled by VPIC\'s unprecedented performance on Roadrunner: modeling laser plasma interaction in upcoming inertial confinement fusion experiments at the National Ignition Facility NIF), modeling short-pulse laser GeV ion acceleration, and modeling reconnection in space and laboratory plasmas.

This work was performed under the auspices of the United States Department of Energy by the Los Alamos National Security LLC Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. Work supported in part by the Laboratory Directed Research and Development (LDRD) Program.

Advances in Kinetic Plasma Simulation with VPIC and Roadrunner



Kevin Bowers*, Brian Albright, Lin Yin, Bill Daughton, Vadim Roytershteyn, Ben Bergen and Tom Kwan
Los Alamos National Lab
* Guest Scientist

Los Alamos

ASC NNSA

Overview

The Software

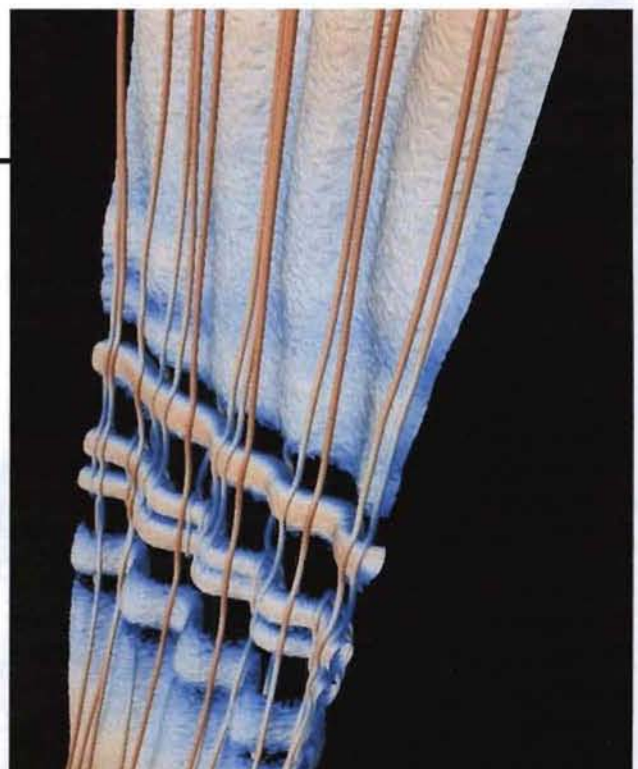
- VPIC: A 3d electromagnetic relativistic particle-in-cell simulation code

The Supercomputer

- Roadrunner: A petascale heterogeneous Cell / Opteron cluster

The Science

- Laser-Plasma Interaction in Inertial Confinement Fusion
- Laser Ion Acceleration
- ~~Magnetic Reconnection~~



Magnetic reconnection simulation in 3d with collisions and open boundary conditions

Los Alamos

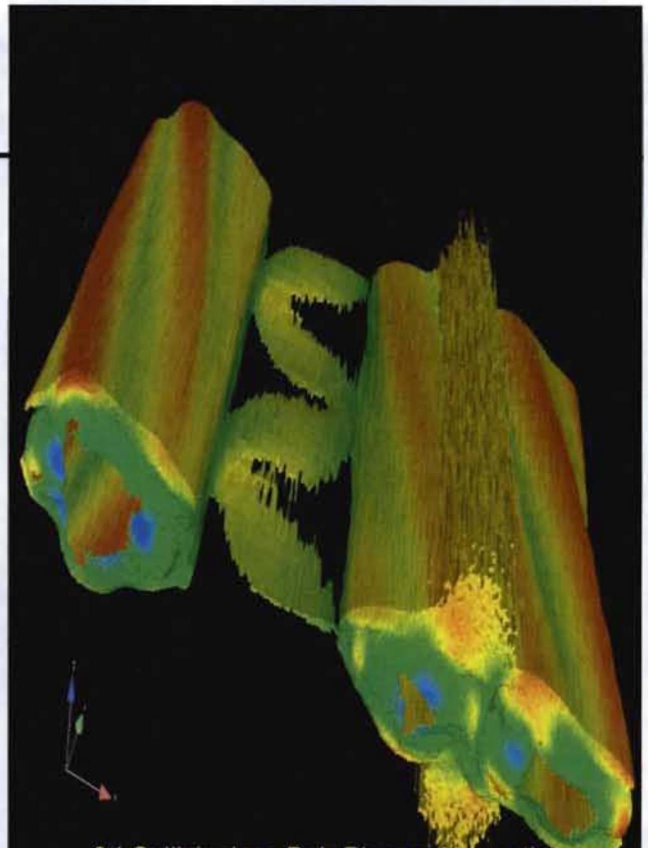
Choir Preaching

Petaflops today

Exaflops in 10 years

Few experimental and observational capabilities will see a comparable increase

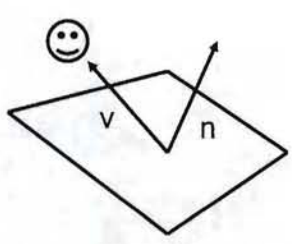
Computational science well positioned for discoveries in biology, chemistry, climate, cosmology, energy, materials, plasmas ...



3d Collisionless Pair-Plasma Magnetic Reconnection (Yin *et al*, Phys Rev Lett, 2008)

Modern CPUs Optimized for Games

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & t_y \\ r_{zx} & r_{zy} & r_{zz} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



$$v \cdot n = |v||n| \cos\theta_{vn}$$

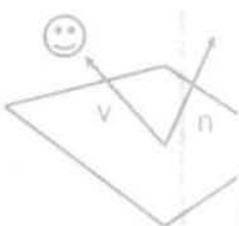
Floating point intensive games use small matrix / short vector ops in single precision

Single precision 4-vector SIMD (Single-Instruction-Multiple-Data) extensions common

Not optimized for traditional double precision large vector operations

Modern CPUs Optimized for Games

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & t_x \\ r_{yx} & r_{yy} & r_{yz} & \\ r_{zx} & r_{zy} & r_{zz} & \\ 0 & 0 & 0 & \end{bmatrix} \begin{bmatrix} x \\ \\ \\ \\ \end{bmatrix}$$



$$v \cdot n = |v||n| \cos \theta_{vn}$$

Roadrunner based on a chip originally designed for the Sony Playstation 3 videogame console

ive games use
ector ops in

ector SIMD
ultiple-Data)

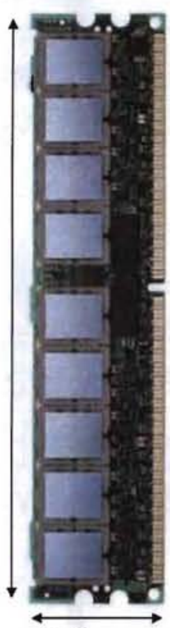
ditional
ge vector
operations

Los Alamos

NNSA

The Speed of Light is Too Slow

13.3 cm



3.0 cm

Consider a registered ECC DDR2-DIMM in a node with 3.2 GHz dual-issue 4-vector SIMD cores (e.g., Roadrunner)

Characteristic time for a signal at the effective speed of light to travel around the DIMM is ~3.2 ns

This alone is ~10 clocks
Time enough for ~80 flops / core

This is optimistic; many other delays

Los Alamos

NNSA

The Speed of Light is Too Slow

Consider a registered ECC DDR2-
Hz dual-
(e.g.,
nal at the
avel
s
s / core
This is optimistic; many other delays

Due to physical limitations, moving data between, and even within, modern microchips is more time consuming than performing computations

You're Smarter Than the Compiler

Languages are far more restrictive than most developers expect. For example, in ANSI C, this optimization is illegal (rightly so---floating point addition is not associative)

```
y = b + c;
z = (a + b) + c;
```

```
y = b + c;
z = a + y;
```

and this FORTRAN-indexed loop cannot be safely unrolled / pipelined / ... in C (why is left as an exercise)

```
for( int i=1; i<=n; i++ ) x[i] += v[i]*dt;
```

while this C-indexed loop can, but only if `x` and `v` are explicitly made restricted pointers

```
for( int i=0; i< n; i++ ) x[i] += v[i]*dt;
```

You're Smarter Than the Compiler

Languages not expressive enough and poorly expose modern HPC capabilities and limitations; compilers lack enough context to optimize well

Computational scientists still need to know something about computation

Situation unlikely to improve; developers unaware of what compilers need and compiler writers unlikely to exploit info anyway (HPC is a moving target niche)

Overview

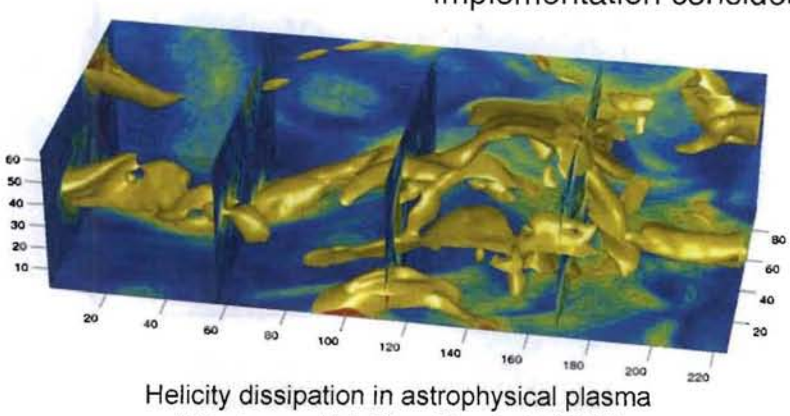
The Software

- VPIC: A 3d electromagnetic relativistic particle-in-cell simulation code

Modeling capabilities

Comparison with other techniques

Implementation considerations



Helicity dissipation in astrophysical plasma
(Bowers and Li, Phys Rev Lett, 2006)

What does VPIC do?

VPIC integrates the relativistic Maxwell-Boltzmann system in a linear background medium for multiple particle species,

$$\partial_t f_s + c\gamma^{-1}\vec{u}\cdot\nabla f_s + \tfrac{q_s}{m_s c}\left(\vec{E} + c\gamma^{-1}\vec{u}\times\vec{B}\right)\cdot\nabla_u f_s = \left(\partial_t f_s\right)_{coll}$$

$$\partial_t\vec{E} = \varepsilon^{-1}\nabla\times\mu^{-1}\vec{B} - \varepsilon^{-1}\vec{J} - \varepsilon^{-1}\sigma\vec{E}$$

$$\partial_t\vec{B} = -\nabla\times\vec{E},$$

in time with an explicit-implicit mixture of velocity Verlet, leapfrog, Boris rotation and exponential differencing based on a reversible phase-space-volume conserving 2nd order Trotter factorization.

Direct discretization of f_s is prohibitive; f_s is sampled by particles,

$$d_t\vec{r}_{s,n} = c\gamma_{s,n}^{-1}\vec{u}_{s,n} \qquad d_t\vec{u}_{s,n} = \tfrac{q_s}{m_s c}\left(\vec{E}\Big|_{\vec{r}_{s,n}} + c\gamma_{s,n}^{-1}\vec{u}_{s,n}\times\vec{B}\Big|_{\vec{r}_{s,n}}\right).$$

Particles obey the same Boltzmann equation outside of collisions.

A smooth J is extrapolated from the particles; as a result, E, B and J can be sampled on a mesh and interpolated to and from particles.

Los Alamos

ASC NNSA

What does VPIC do?

VPIC integrates the relativistic Maxwell-Boltzmann system in a linear background medium for multiple particle species,

$$\partial_t f_s + c\gamma^{-1}\vec{u}\cdot\nabla f_s + \ldots$$

$$\partial_t\vec{E} = \varepsilon^{-1}\nabla\times\mu^{-1}\vec{B} - \ldots$$

$$\partial_t\vec{B} = -\nabla\times\vec{E},$$

Theoretical details useful for making babies cry

(Actually, the slide appendix has a detailed theoretical methods overview for the brave)

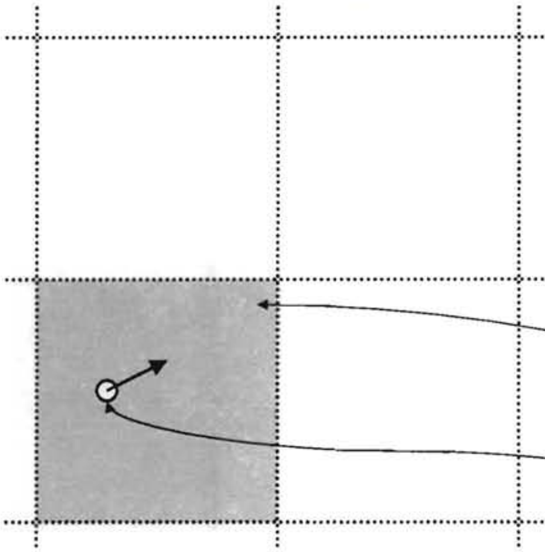
$$d_t\vec{r}_{s,n} = c\gamma_{s,n}^{-1}\vec{u}_{s,n}$$

A smooth J is extrapolated from the particles; as a result, E, B and J can be sampled on a mesh and interpolated to and from particles.

Los Alamos

ASC NNSA

What does VPIC really do?



Initial State

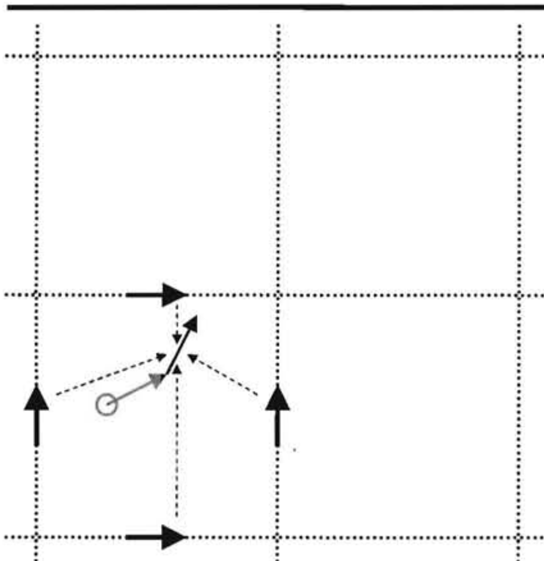
PIC: Particle In Cell
(a.k.a. Voxel)

Read:	32 bytes
Write:	0 bytes
Compute:	0 flop

Los Alamos

ASC NNSA

What does VPIC really do?



Initial State

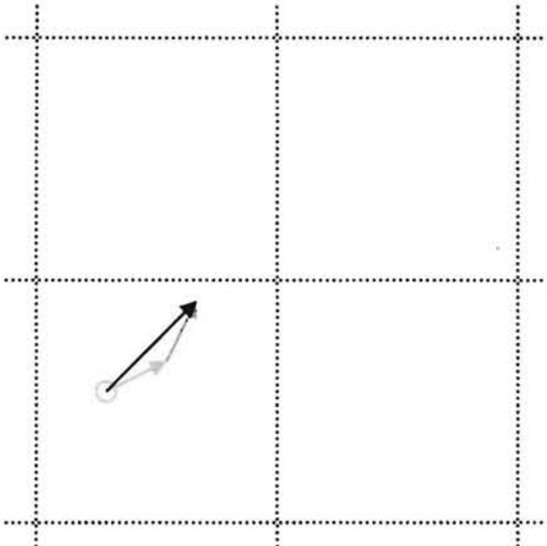
Interpolate E and B

Read:	72 bytes
Write:	0 bytes
Compute:	27 flop

Los Alamos

ASC NNSA

What does VPIC really do?



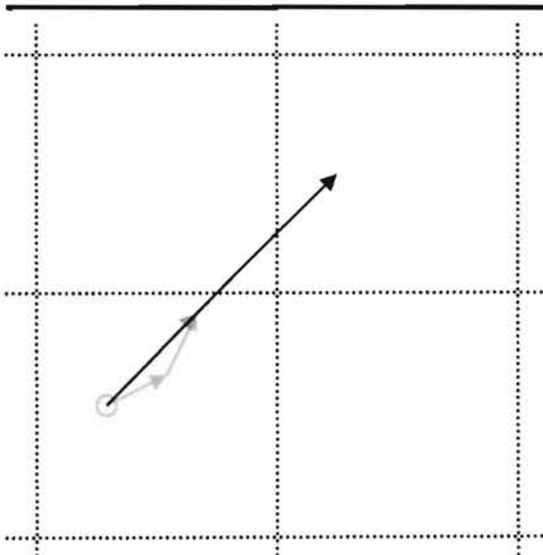
Initial State

Interpolate E and B

Update u

Read: 0 bytes
Write: 0 bytes
Compute: 107 flop

What does VPIC really do?



Initial State

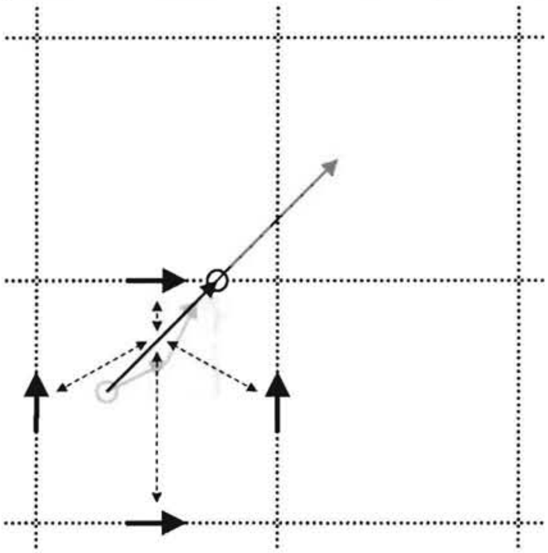
Interpolate E and B

Update u

Compute Motion

Read: 0+48 bytes
Write: 0+48 bytes
Compute: 42+70 flop

What does VPIC really do?



Initial State

Interpolate E and B

Update u

Compute Motion

Update r and J

Read: 56 bytes

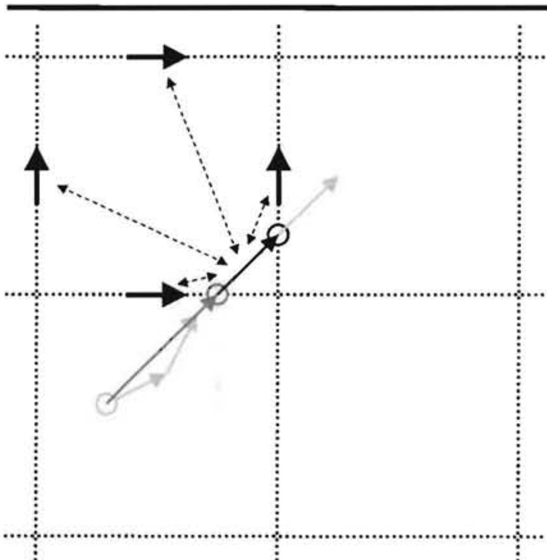
Write: 48 bytes

Compute: 168 flop

Los Alamos

ASC NNSA

What does VPIC really do?



Initial State

Interpolate E and B

Update u

Compute Motion

Update r and J

Update r and J

Read: 56 bytes

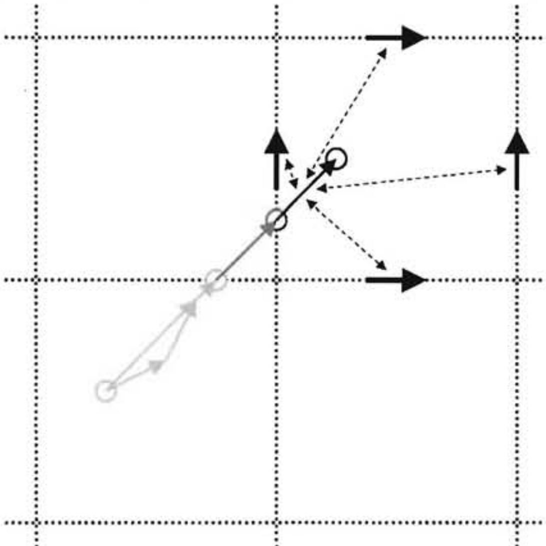
Write: 48 bytes

Compute: 168 flop

Los Alamos

ASC NNSA

What does VPIC really do?



Initial State

Interpolate E and B

Update u

Compute Motion

Update r and J

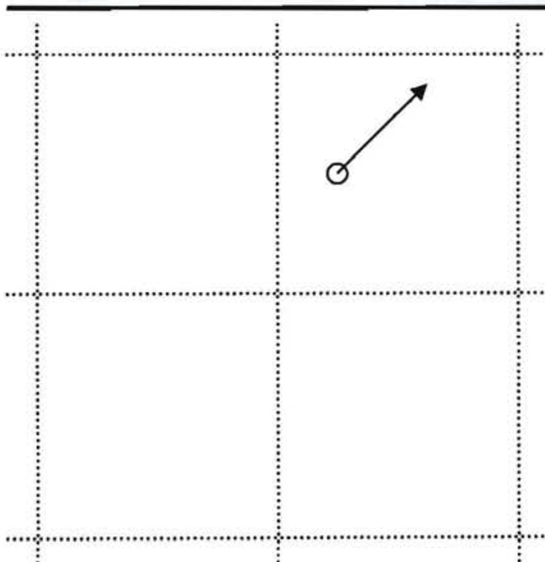
Update r and J

Update r and J

Read:	56 bytes
Write:	48 bytes
Compute:	168 flop

Los Alamos

What does VPIC really do?



Initial State

Interpolate E and B

Update u

Compute Motion

Update r and J

Update r and J

Update r and J

Final State

Read:	0 bytes	**Net Read:**	152+ 56 n_c bytes
Write:	32 bytes	**Net Write:**	80+ 48 n_c bytes
Compute:	0 flop	**Net Compute:**	246+168 n_c flop

Los Alamos

Why use PIC?

Vlasov codes model similar equations

- But do not scale to high dimensional systems

Why use PIC?

Vlasov codes model similar equations

- But do not scale to high dimensional systems

Traditional Monte-Carlo easy to parallelize + accelerate

- But not suitable for time dependent effects

Why use PIC?

Vlasov codes model similar equations

- But do not scale to high dimensional systems

Traditional Monte-Carlo easy to parallelize + accelerate

- But not suitable for time dependent effects

Computational fluid dynamics cheaper

- But impossible if the equation of state is unknown

Why use PIC?

Vlasov codes model similar equations

But do not scale to high dimensional systems

Traditional Monte-Carlo easy to parallelize + accelerate

- But not suitable for time dependent effects

Computational fluid dynamics cheaper

- But impossible if the equation of state is unknown

Molecular dynamics closely related

- But orders of magnitude more expensive ...

MD versus PIC

MD focus is short range

- Necessary when nearby interaction potential energy >> thermal energy
- Difficult for particles to represent many atoms
- ***Flops / particle / step large (10^3 - 10^4)***

MD versus PIC

MD focus is short range

- Necessary when nearby interaction potential energy >> thermal energy
- Difficult for particles to represent many atoms
- ***Flops / particle / step large (10^3 - 10^4)***

PIC focus is long range

- Useful when nearby interaction potential energy << thermal energy
- Approximates short range interactions
- ***Flops / particle / step small ($\sim 10^2$)***

Typical VPIC Simulations

Many particles / node (10^7 - 10^8)

- Particle data does not fit in cache
- >90% expense is particle pushing

Typical VPIC Simulations

Many particles / node (10^7 - 10^8)

- Particle data does not fit in cache
- >90% expense is particle pushing

Many voxels / node (10^4 - 10^5)

- Field data does not fit in cache
- Many particles / voxel (10^2 - 10^4)

Typical VPIC Simulations

Many particles / node (10^7 - 10^8)

- Particle data does not fit in cache
- >90% expense is particle pushing

Many voxels / node (10^4 - 10^5)

- Field data does not fit in cache
- Many particles / voxel (10^2 - 10^4)

Few voxel boundaries crossed / particle / step

- Speed of light well resolved and $v<c$

Typical VPIC Simulations

Many particles / node (10^7 - 10^8)

- Particle data does not fit in cache
- >90% expense is particle pushing

Many voxels / node (10^4 - 10^5)

- Field data does not fit in cache
- Many particles / voxel (10^2 - 10^4)

Few voxel boundaries crossed / particle / step

- Speed of light well resolved and $v<c$

Internode communications naturally optimal

- Communication every step, but, because of finite c, data needed on a node already there or nearby

Typical VPIC Simulations

Many p
- Parti
- >90%

Many v
- Field
- Many

Few vo
- Spee

Internod
Com … ite *c*,
data … y

VPIC isn't like a matrix calculation with $O(N^3)$ compute on $O(N^2)$ data

Low compute to data motion ratio (~1 flop / byte) makes high performance hard to achieve

Performance limited by local data motion

Los Alamos — ASC NNSA

Implementation rules of thumb

Operation		Time	Rel Cost
Data	Internode	10 μs	100,000
Access	Memory	50 ns	500
(Latency)	L2 cache	5 ns	50
	L1 cache	1 ns	10
Data	Internode	5 ns	50
Movement	Memory	0.5 ns	5
(32-bit)	L2 cache	0.2 ns	2
	L1 cache	0.1 ns	1
Sing Prec	FLOP	0.1 ns	1

Los Alamos — ASC NNSA

Implementation rules of thumb

Minimize data access, data movement and computation, in that order

The ratio between computation and data motion costs (particularly latency) likely to get even worse

Computation and storage are virtually free compared to data motion; replicating computations and data often worthwhile.

Bad Ideas

Absolute particle coordinates	***Destroys precision*** Bits wasted resolving voxel indices ***Slow interpolation*** Float - int casts (or worse)

Bad Ideas

Absolute particle coordinates	***Destroys precision*** Bits wasted resolving voxel indices
	Slow interpolation Float - int casts (or worse)
Unsorted particles	***Cache misses*** Field data accessed randomly

Bad Ideas

Absolute particle coordinates	***Destroys precision*** Bits wasted resolving voxel indices
	Slow interpolation Float - int casts (or worse)
Unsorted particles	***Cache misses*** Field data accessed randomly
Advance done with several passes	***Bandwidth wasted*** Data touched several times / step

Bad Ideas

Absolute particle coordinates	*Destroys precision* Bits wasted resolving voxel indices *Slow interpolation* Float - int casts (or worse)
Unsorted particles	*Cache misses* Field data accessed randomly
Advance done with several passes	***Bandwidth wasted*** Data touched several times / step
Each component stored in own array	***Bandwidth wasted*** Small unaligned accesses

Los Alamos

ASC NNSA

Bad Ideas

Absolute particle coordinates	*Destroys precision* Bits wasted resolving voxel indices *Slow interpolation* Float - int casts (or worse)
Unsorted particles	*Cache misses* Field data accessed randomly
Advance done with several passes	*Bandwidth wasted* Data touched several times / step
Each component stored in own array	***Bandwidth wasted*** Small unaligned accesses

Field samples used for interpolation	***Too few "ways" to keep track*** 29 diff memory regions accessed / particle

Los Alamos

ASC NNSA

Bad Ideas

If VPIC were implemented conventionally, ~31 physical DRAM transfers / particle / step and not many flops to show for them

Need data flow optimization techniques

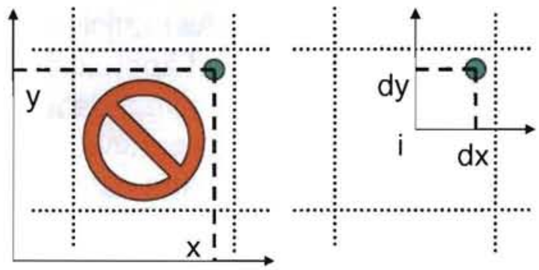
Scientific codes often use data structures that are easy to implement quickly but limit flexibility and scalability in the long run

Good Ideas

Voxel index + offset particle coordinates	***Maximizes precision*** Bits conserved; critical in single precision
	Fast interpolation No casts; almost trivial computation
Sorted particles	***Cache hits*** Field data approximately streamed
Advance done in a single pass	***Bandwidth conserved*** Particle data touched once / step
Similar components grouped together	***Bandwidth conserved*** Large aligned accesses
Precompute voxel interpolation coeffs	***Many "ways" to keep track*** 2 diff memory regions accessed / particle

Position representation

Positions are given by a voxel index and the offset from the voxel center, normalized to the voxel dimensions



Determining which interpolation coefficients to load is trivial

Field interpolation and current accumulation can use particle offsets directly

Generalizes naturally to other methods (e.g., irregular meshes)

Requires transforming the offset coordinate system whenever a particle crosses a voxel boundary. Cost is trivial and elegantly incorporated into the charge conserving current accumulation

Position representation (cont)

When an absolute position representation is used, some position bits encode the voxel index, leaving fewer bits to encode the offset. Consider a single precision 1d simulation with $2^{10} = 1024$ voxels over the domain *(0,1)*:

Absolute coordinates

Particles in voxel *0* see a $2^{-24}/2^{10} \sim 6e\text{-}11$ worst case absolute resolution while those in voxels *512-1023* see a $2^{-24} \sim 6e\text{-}8$ resolution

Numerical anisotropy from position resolution varies by orders of magnitude over the domain

Index+offset coordinates

Regardless of voxel, all particles see a $2^{-25}/2^{10} \sim 3e\text{-}11$ resolution (the sign bit provides an extra bit)

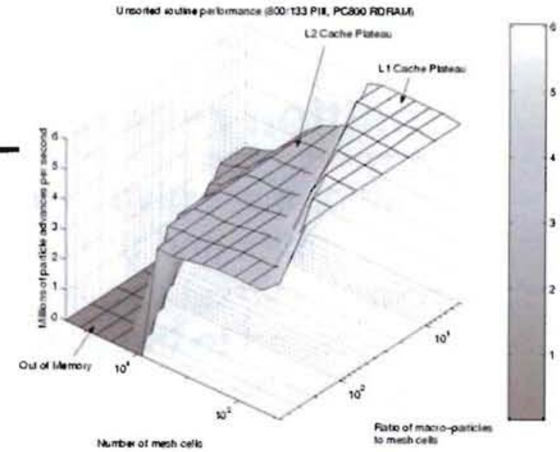
Better than absolute coordinates everywhere (by orders of magnitude in most places) with no numerical anisotropy

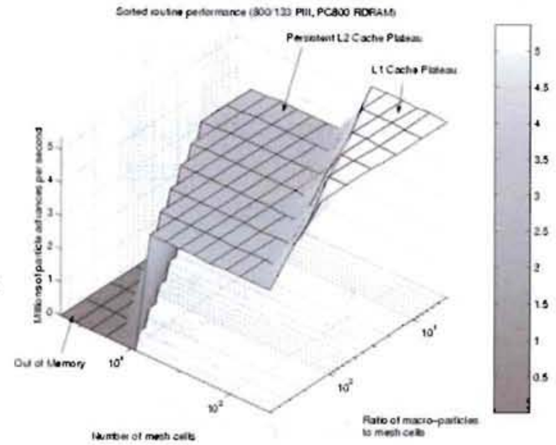
Particle sorting

Particles are periodically sorted by their voxel index. All particles in a voxel are processed approximately sequentially and the field data necessary for these particles loaded once from memory and cached

Sorting is infrequent (10s of steps) but done rapidly using a NUMA-friendly thread-parallel version of Bowers JCP 2001 (See SciDAC09 paper for details)

Allows various collision models to be implemented efficiently





Los Alamos

Single pass processing and particle data layout

Performance asymptotically limited by number of times a particle is moved between CPU and DRAM per step on average. Single pass processing ideal:

```
for each particle,
  interpolate E and B
  update u and compute movement
  update r and accumulate J
  if an exceptional boundary hit,
    save particle index and
      remaining movement
  end if
end for
```

Particle data is stored contiguously, aligned and organized for 4-vector SIMD. The above loop thus streams through particles using large aligned transfers under the hood—the ideal access pattern

```
typedef struct {
  float dx, dy, dz; int i; // Cell offset (on [-1,1]) and index
  float ux, uy, uz, q;     // Normalized momentum and charge
} particle_t;
```

Los Alamos ASC NNSA

Field interpolation

For each voxel, interpolation coefficients are precomputed before the particle advance and saved in a contiguous, aligned, 4-vector SIMD compatible layout:

```
typedef struct {
  float ex, dexdy, dexdz, d2exdydz;
  float ey, deydz, deydx, d2eydzdx;
  float ez, dezdx, dezdy, d2ezdxdy;
  float bx, dbxdx, by,    dbydy;
  float bz, dbzdz, pad0,  pad1;
} interpolator_t;
```

Because particles are sorted, coefficients are accessed approximately sequentially in large aligned transfers a near minimal number of times

Even though the coefficients require over *3* times more storage than the raw fields, the net impact is to reduce memory transfers to minimal levels by making more efficient use of cache

Los Alamos

NNSA

Current accumulation

Determining the voxels through which a particle passed varies from particle to particle; one particle might remain in the cell in which it started while the next might cross through several. To utilize SIMD, VPIC exploits that particles do not cross voxel boundaries often.

VPIC advances 4 particles at a time with 4-way SIMD by assuming none of the 4 particles cross voxel boundaries. Particles that do cross are detected and make no current contributions during this process. These particles are processed in scalar code subsequently

Like the interpolation coefficients, current contributions from particle motion in a voxel are made to a contiguous aligned set of partial currents. These are post-processed into *J* prior to the field advance. The same benefits described for field interpolation apply

Los Alamos

NNSA

Exceptions

If a particle hits an "exceptional" boundary (e.g. needs communication to a neighboring node, needs absorbed, needs refluxed, ...) during voxel crossing current accumulation, the index and remaining particle movement are saved to an exception list for later processing.

No additional passes through the particles are necessary; exception records are streamed to memory

Slow application specific code cleanly separated from the high performance general particle advance

Exception handling does not pollute the caches while the particle advance is running.

4-way SIMD

Languages are not expressive enough to allow compilers to use 4-way SIMD in operations as complex as those in VPIC

VPIC implements a language extension that allows C-style code to converted automatically to high performance platform specific 4-way SIMD instructions with low overhead. A similar approach used in Bowers *et al* Supercomputing 2006.

```
// Interpolate ex for the next 4 particles
load_4x4_tr( interp_coeff[ i(0) ].QUAD( ex, dexdy, dexdz, d2exdydz ),
             interp_coeff[ i(1) ].QUAD( ex, dexdy, dexdz, d2exdydz ),
             interp_coeff[ i(2) ].QUAD( ex, dexdy, dexdz, d2exdydz ),
             interp_coeff[ i(3) ].QUAD( ex, dexdy, dexdz, d2exdydz ),
             ex, dexdy, dexdz, d2exdydz );
ex = (ex + dy*dexdy) + dz*(dexdz + dy*d2exdydz);
```

No Apologies

VPIC designed with single precision in mind

- Half bytes moved and wider SIMD available

No Apologies

VPIC designed with single precision in mind

- Half bytes moved and wider SIMD available

Usually, discretization error >> single precision error

- Single precision okay if very carefully implemented
- Doubles and "numerical hygiene" used as necessary
- Extensive convergence studies and validation against theory, experiment, double precision codes

No Apologies

VPIC designed with single precision in mind

- Half bytes moved and wider SIMD available

Usually, discretization error >> single precision error

- Single precision okay if very carefully implemented
- Doubles and "numerical hygiene" used as necessary
- Extensive convergence studies and validation against theory, experiment, double precision codes

Stabilized to the point where each voxel has identical numerical properties regardless how the voxel mesh is translated, oriented or reflected

Los Alamos

ASC NNSA

No Apologies

VPIC d

- Half b

Usually ... rror

- Singl ... nted
- Doub ... ically
- Exter ... agai ... odes

Sta ... has identi ... w the voxe ... cted

When in single precision, developers care <u>more</u> about arithmetic error

Unlike double precision, ignoring it often leads to catastrophes

We die a little bit on the inside when CPUs and compilers take short cuts (they often do)

Los Alamos

ASC NNSA

Overview

The Supercomputer

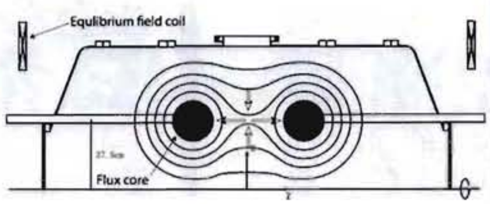
- Roadrunner: A petascale heterogeneous Cell / Opteron cluster

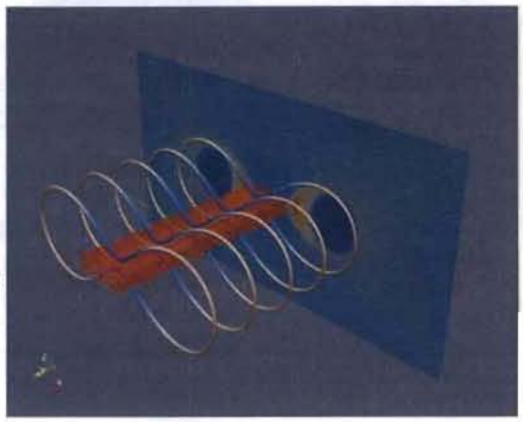
Hardware Description

Porting Details

Measured performance

Preliminary 3d Collisional VPIC Simulation of MRX (Magnetic Reconnection eXperiment)

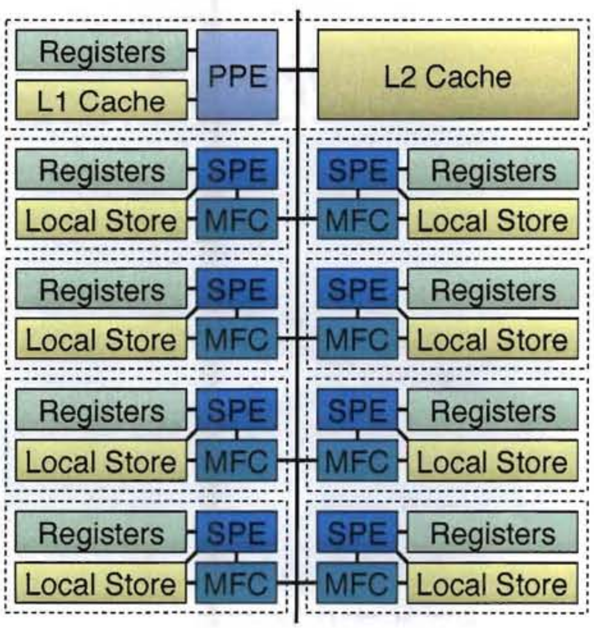




Los Alamos

NNSA

Cell Broadband Engine



1 general purpose core, "PPE"

8 special 4-vector SIMD cores, "SPE"

Each SPE can only directly access its 256KB "local store"

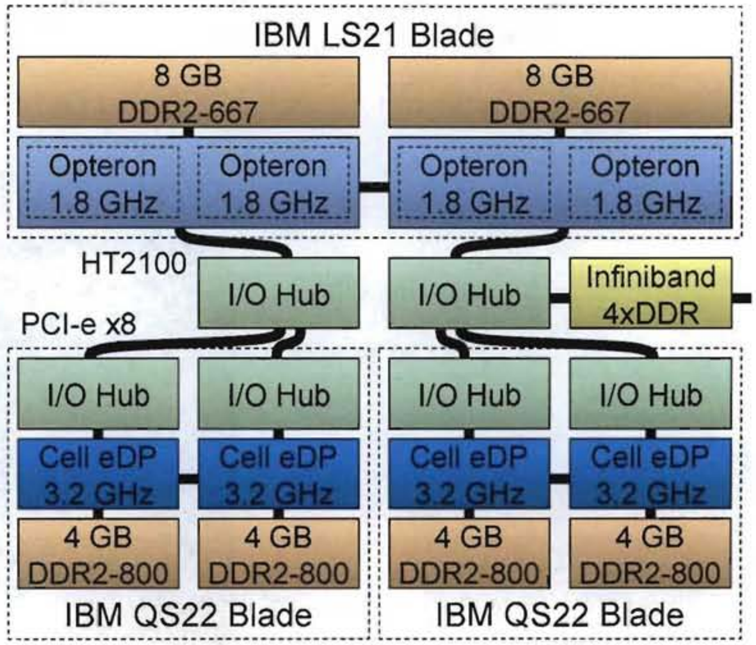
Local store like cache but memory transfers explicitly managed by "MFC"

Los Alamos

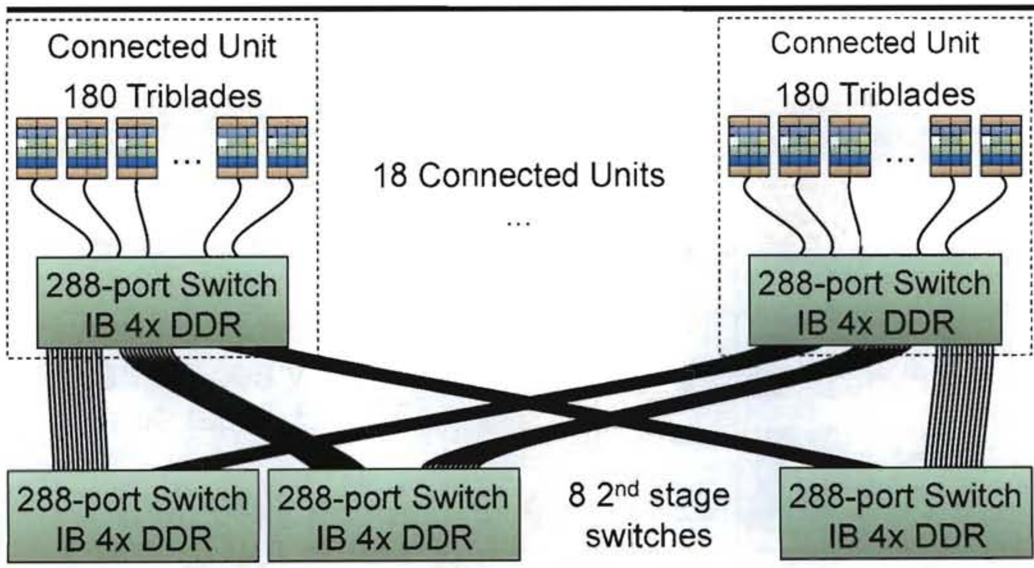
NNSA

Triblade Compute Nodes

Opteron cores one-to-one with Cell eDPs (2 GB/s bandwidth)



Roadrunner



12,960 Opteron cores - ***0.1 Pflop/s (s.p.)***

12,960 Cell eDP chips - ***3.0 Pflop/s (s.p.)***

Porting

Observations

- Most compute in the SPEs
- SPE / Cell DRAM bandwidth (25 GB/s) >>
  SPE / Opteron DRAM bandwidth (2 GB/s)
- Bandwidth off-node same for Cell and Opteron (IB)

Porting

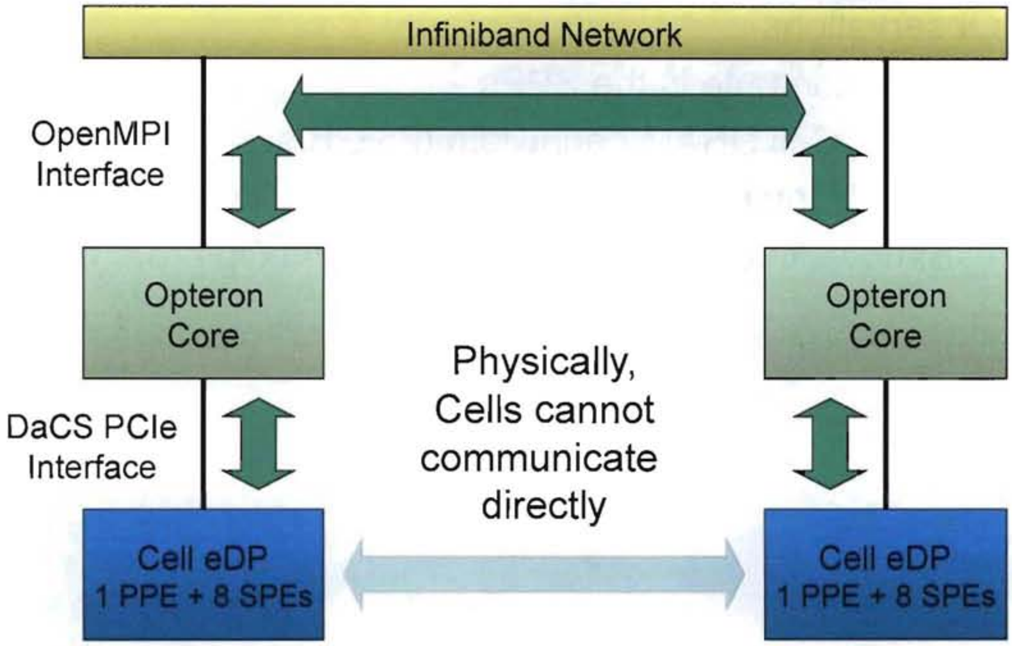
Observations

- Most compute in the SPEs
- SPE / Cell DRAM bandwidth (25 GB/s) >>
  SPE / Opteron DRAM bandwidth (2 GB/s)
- Bandwidth off-node same for Cell and Opteron (IB)

Strategy: **Flatten Roadrunner**

- All calculations done on Cells
- All data stored in Cell DRAM
- Opterons relay Cell communication and I/O

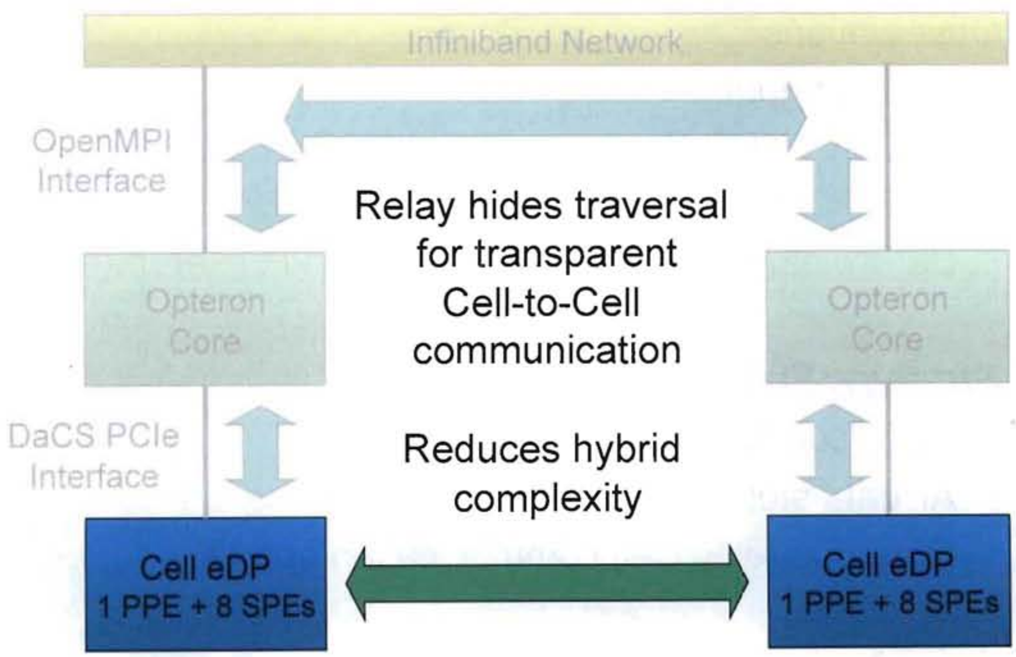
Relay Library



Los Alamos

NNSA

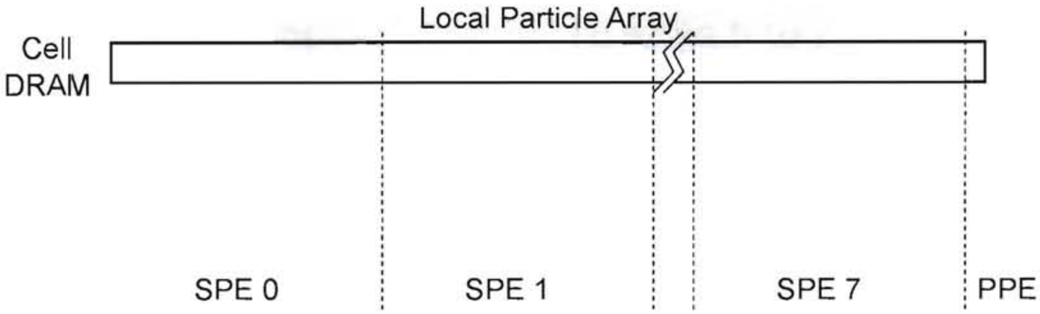
Relay Library



Los Alamos

NNSA

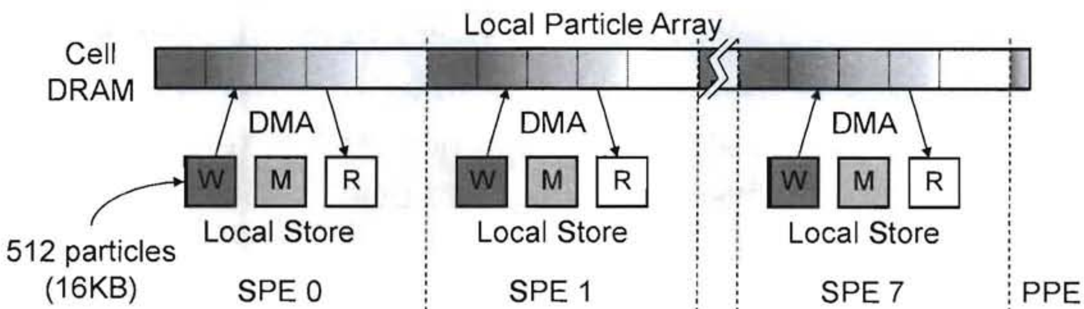
SPE Accelerated Particle Advance



Each SPE assigned a segment containing a multiple of 16 particles and an exclusive current accumulator

The PPE assigned leftover particles

SPE Accelerated Particle Advance



Each SPE assigned a segment containing a multiple of 16 particles and an exclusive current accumulator

The PPE assigned leftover particles

SPEs stream through segments with triple buffering in blocks of 512 particles

SPE Accelerated Particle Advance

The heart of it all: A 512-line part read-only / part write-back software cache handles random access

- **Fully-associative:** A line can hold any voxel's data
- **Least-recently-used:** New data evicts oldest data

The last 512 unique requests <u>guaranteed</u> in cache

SPE Accelerated Particle Advance

The heart of it all: A 512-line part read-only / part write-back software cache handles random access

- **Fully-associative:** A line can hold any voxel's data
- **Least-recently-used:** New data evicts oldest data

The last 512 unique requests <u>guaranteed</u> in cache

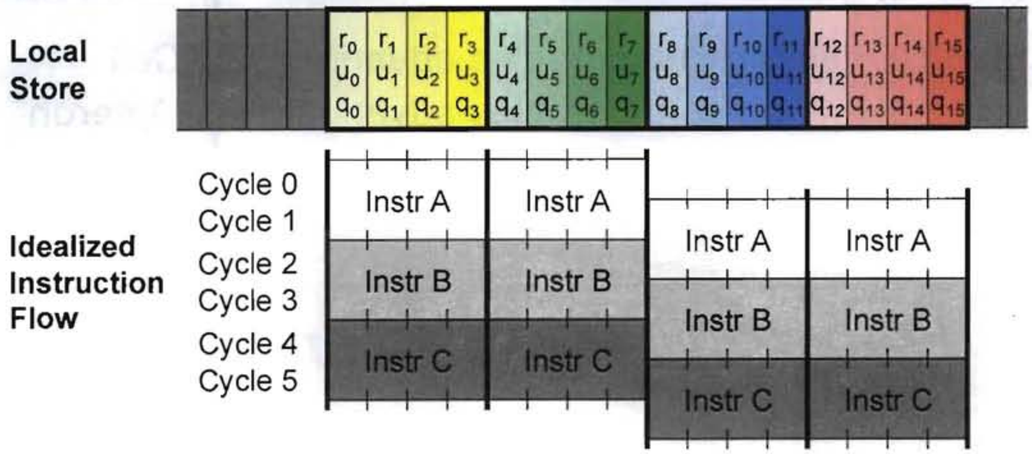
`cache_fetch` called on all 512 particles in a new block

- Most are hits; DMA transfers started for misses
- Returns which lines will hold the voxels' data

`cache_wait` then completes any pending fetches

`cache_fetch` non-trivial internally but a fast *O*(1)

SPE Accelerated Particle Advance



Particles processed 16 at a time

- Original x86 4-vector SIMD kernel hand unrolled and modulo scheduled by 4; register file size (128), pipeline hazards and local store limit further unrolling

Los Alamos

SPE Accelerated Particle Advance

All these techniques result in:

A SPE kernel that operates exclusively out of local store

Most SPE registers used
Most local store used
All 32 DMA channels / SPE used
Most DMA transfers overlapped
Many independent SIMD instructions
Minimal DMA transfers / particle

Los Alamos

Kernel Performance

162.0 million cold particles advanced / s / Cell
÷ 10.3 million cold particles advanced / s / Opteron

15.7x speedup

Kernel Performance

162.0 million cold particles advanced / s / Cell
÷ 10.3 million cold particles advanced / s / Opteron

15.7x speedup
÷ **1.8x** faster SPE clock rate
÷ **8.0x** more SPE cores than Opteron cores

1.1x clock-for-clock speedup, in spite of SPE minimalism and VPIC's tuning for x86

Kernel Performance

162.0 million cold particles advanced / s / Cell
÷ 10.3 million cold particles advanced / s / Opteron

15.7x speedup
÷ **1.8x** faster SPE clock rate
÷ **8.0x** more SPE cores than Opteron cores

1.1x clock-for-clock speedup, in spite of SPE minimalism and VPIC's tuning for x86

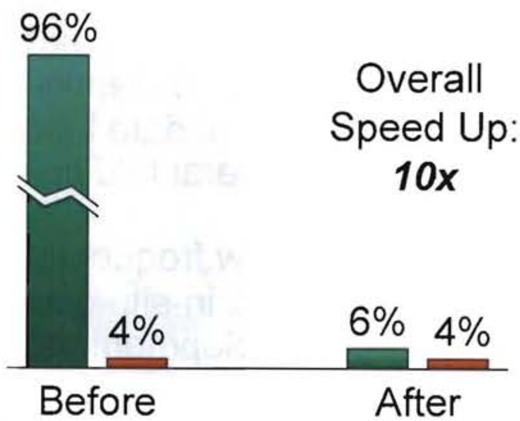
0.517 *Pflop/s on all* 18 *Roadrunner Connected Units*

Need 203,000 Opteron cores for similar performance

Amdahl's Whack-a-Mole

Particle advance accelerated 15.7x

Amdahl's Law:
Rest of code relatively more costly



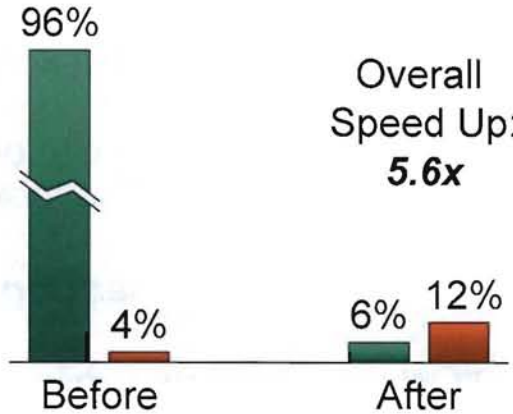
Amdahl's Whack-a-Mole

Particle advance accelerated 15.7x

Amdahl's Street Justice:

Rest of code <u>absolutely</u> more costly

PPE cores less powerful than Opteron cores



Amdahl's Whack-a-Mole

Particle advance accelerated 15.7x

Amdahl's Street Justice:

Rest of code <u>absolutely</u> more costly

PPE cores less powerful than Opteron cores

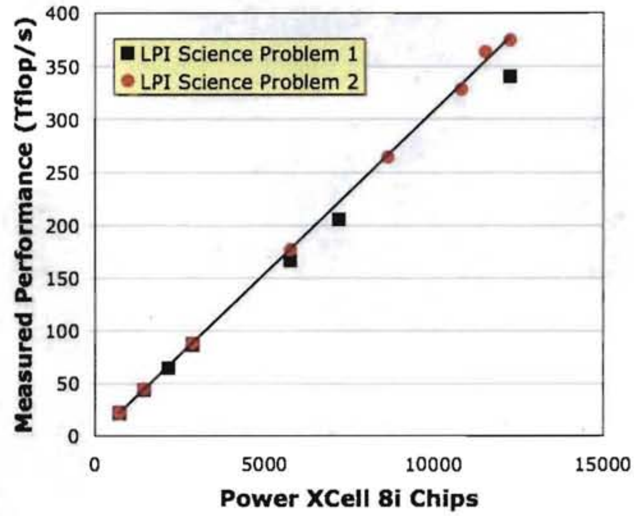
End-to-end performance more sensitive to unaccelerated kernels than conventional platforms. Particle sort and many field update kernels were also SPE accelerated (several fold speedups).

Amdahl bottlenecks are now frequently one-off user-provided application-specific in-situ diagnostics. User experience, improved development models needed.

End-to-End Performance

Two simulations in LPI parameter study (Albright *et al*, Phys Plasmas, 2008) used to benchmark weak scaling

Same physics but 10x faster



Trillion-particle simulations at* 0.374 *Pflop/s sustained on* 17 *CUs (Bowers* et al*, SC08)

Los Alamos

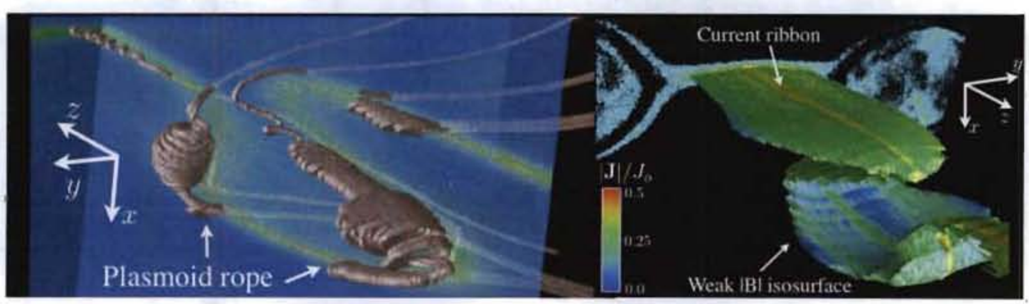
NNSA

Overview

The Science
- Laser-Plasma Interaction in Inertial Confinement Fusion
- Laser Ion Acceleration
- ~~Magnetic Reconnection~~

For each, a brief overview of current research with VPIC on Roadrunner

Conclusions

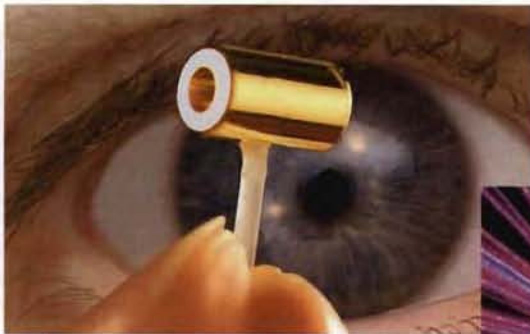


Magnetic Island Detachment (Yin *et al*, Phys Rev Lett, 2008)

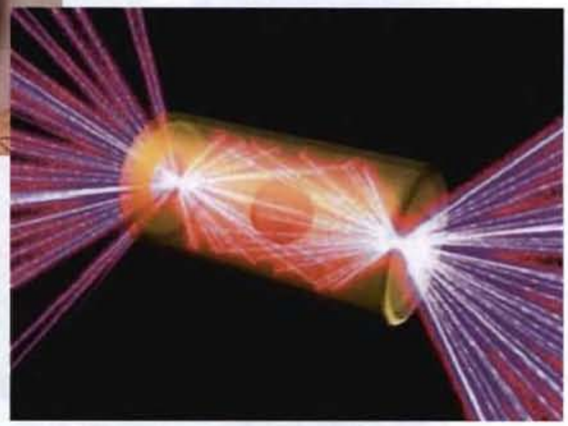
Los Alamos

NNSA

Inertial Confinement Fusion



Lasers implode a fusion fuel capsule to "ignite" it; thermonuclear burning plasma

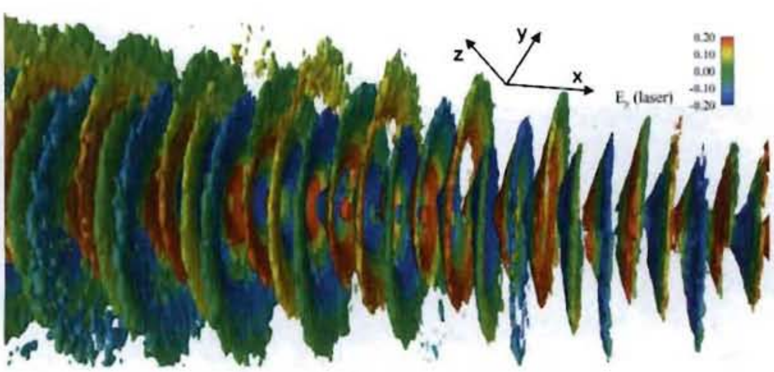


"Minimizing laser-plasma instabilities in the NIF hohlraum is a key to achieving ignition."
- LLNL web site

Inertial Confinement Fusion

LPI (Laser Plasma Interaction) an issue

- **Laser scattering:** Too little compression
- **Laser scattering:** Asymmetric compression
- **e^- Preheating:** Harder to compress hot plasma



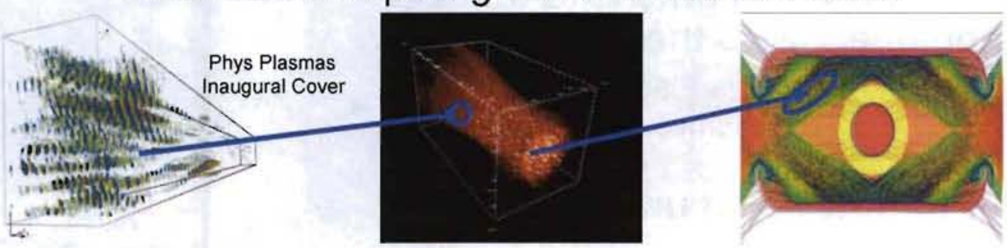
LPI Nonlinear Saturation (Yin *et al*, Phys Rev Lett, 2007)

The Petascale Challenge

In 2010, ICF ignition experiments start
at Livermore's National Ignition Facility (NIF)

The multi-billion dollar question:
What is the risk from LPI?

Petascale computing can address this issue



LANL VPIC LPI modeling | **LLNL pF3D laser modeling** | **LLNL Hydra ICF modeling**

Los Alamos

ASC NNSA

Computational Science in Action

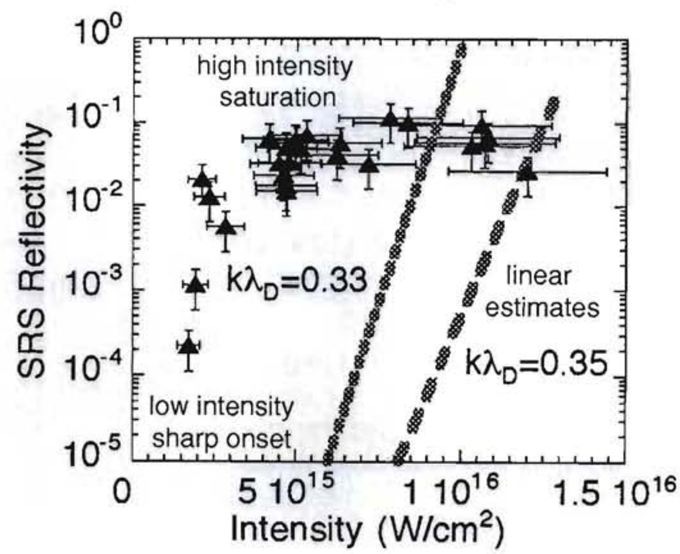
Linear theory for SRS (Stimulated Raman Scattering) in LPI developed

Drake *et al*, Phys Fluids, 1974

Trident experiments observe unexplained behavior

Montgomery *et al*, Phys Plasmas, 2002

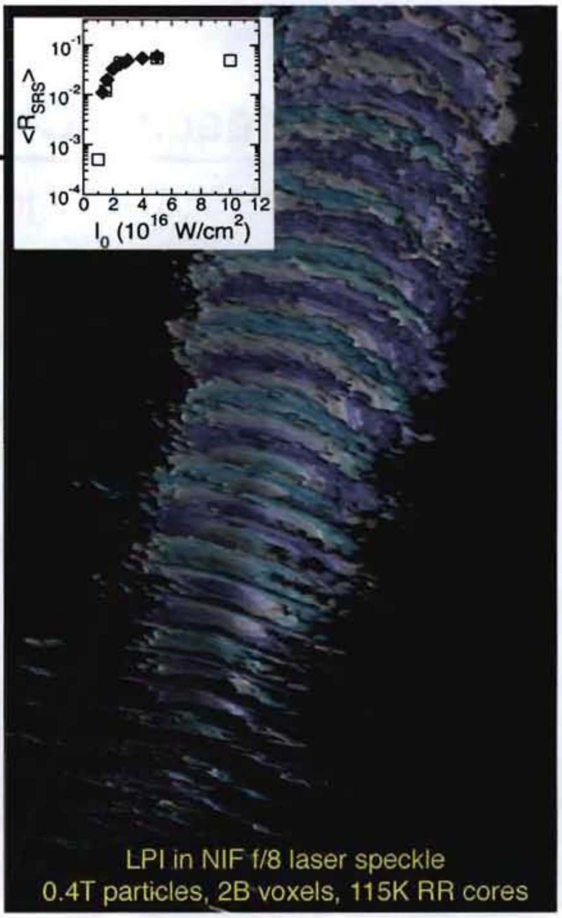
Trident Experiments (527 nm, f/4.5
Gaussian beam, T_e=700eV)



Los Alamos

ASC NNSA

Computational Science in Action



VPIC identifies key physics

Plasma wave bowing, self-focusing, filamentation and trapped particle modulational instability cause rapid onset and saturation (Yin *et al*, Phys Rev Lett, 2007)

Reflectivity agrees with experiment

Simulation insights lead to non-linear SRS theories

Rose and Yin, Phys Plasmas, 2008, Yin *et al*, Phys Plasmas, 2009

VPIC now being used on Roadrunner to understand and predict LPI in NIF

Los Alamos

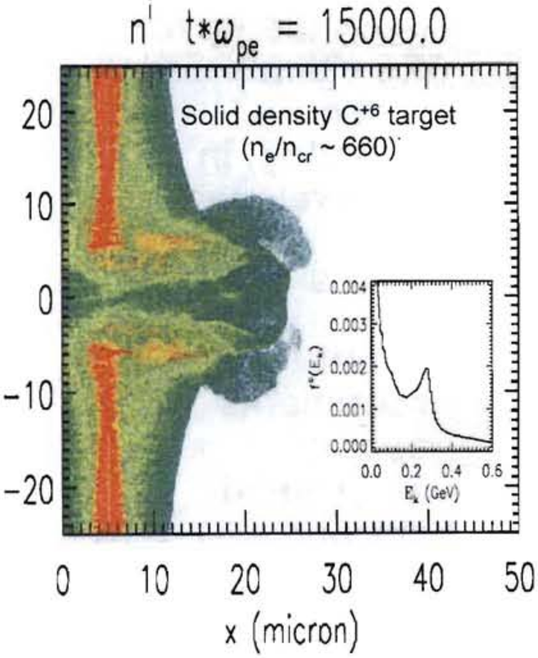
Laser Ion Acceleration

High energy C^{+6} beams observed from an ultra-intense short laser pulse incident on a thin foil

Via target normal sheath acceleration process (Hegelich *et al*, Nature, 2006, Albright *et al*, Phys Rev Lett, 2006)

VPIC corroborates and discovers a process for higher energies

Relativistic effects make foil transparent for ultra-high contrast pulses and thinner foils, allowing pulse to "breakout" and accelerate ions (Yin *et al*, Laser and Particle Beams, 2006)



Los Alamos

ASC NNSA

Laser Ion Acceleration

Simulation insights lead to new acceleration theories

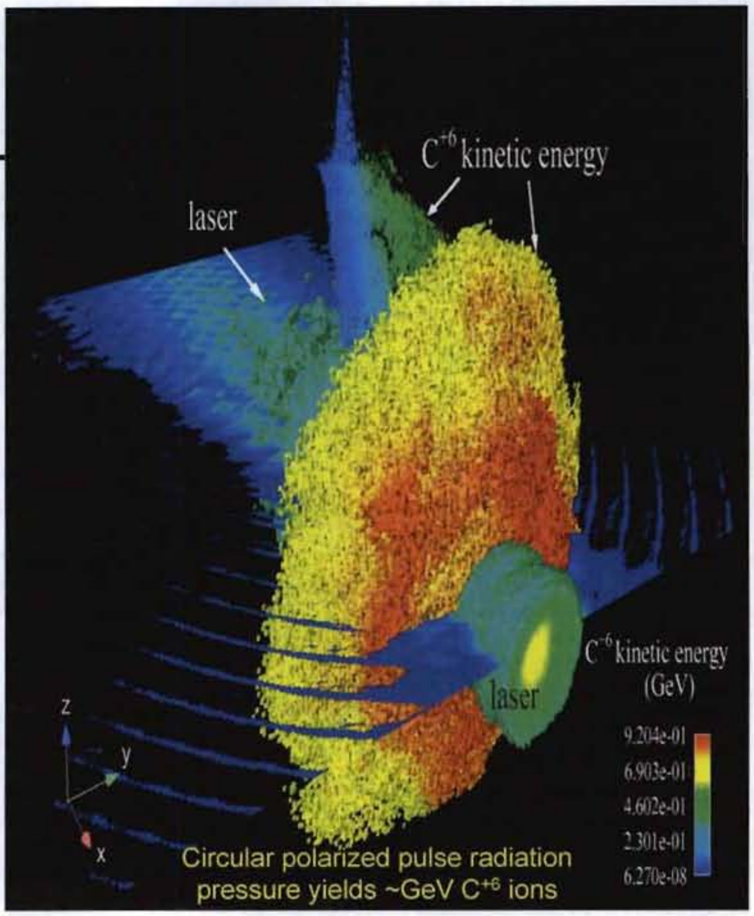
Relativistic Buneman instability for linear polarization (Albright *et al*, Phys Plasmas 2007)

VPIC prediction experimentally confirmed

Prediction drove Trident's redesign

Henig *et al*, Phys Rev Lett, 2009 (in press)

Los Alamos



Conclusions

Petascale supercomputers can change the way we do science

Tapping the potential requires rethinking codes and analysis

Data motion is not free

Supercomputers getting faster but not the speed of light

Data flow optimization future proofs codes

VPIC data flow optimized almost 8 years ago yet needed no structural modifications to realize order-of-magnitude speedups on Roadrunner

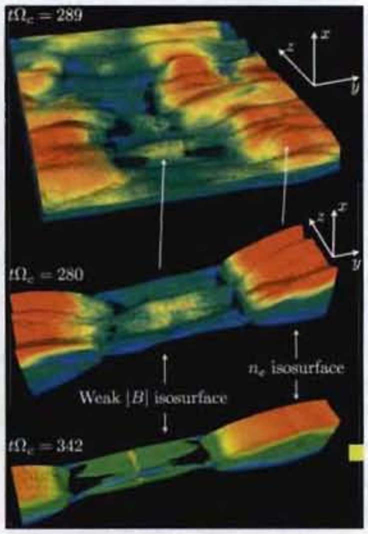
Roadrunner is a glimpse of the future

Routine petascale computations, 100,000+ core parallelism, heterogeneous cores and intermingled compute / memory

Data flow optimization paramount

NNSA Los Alamos IBM

Acknowledgments



Harris sheet tearing (Yin *et al*, Phys Rev Lett, 2008)

Research supported in part by the Los Alamos LDRD Program, DOE, NSF and NASA

Special thanks to IBM Roadrunner team (Cornell Wright, Bill Brandmeyer and Chris Engel) for the opportunity to use Roadrunner during early testing

Thanks to Drs. Ken Koch, Hui Li, Jeremy Margulies, Eric Nelson and Tiankai Tu for assistance with slides. Most 3d visualizations performed with EnSight Gold by CEI Inc

Work performed under the auspices of the United States Department of Energy by the Los Alamos National Security LLC Los Alamos National Laboratory under contract DE-AC52-06NA25396

Los Alamos

Appendix

Los Alamos

Operator splitting

For a well-behaved operator, the operator equation

$$d_t X = \hat{L} X$$

has the formal solution

$$X(t + \delta_t) = e^{\delta_t \hat{L}} X(t)$$

If L can be split into N well-behaved operators

$$\hat{L} = \sum_{i=1}^{N} \hat{L}_i$$

a 2[nd] order approximation of the operator exponential is:

$$e^{\delta_t \hat{L}} \sim e^{\delta_t \hat{L}_1/2} \ldots e^{\delta_t \hat{L}_{N-1}/2} e^{\delta_t \hat{L}_N} e^{\delta_t \hat{L}_{N-1}/2} \ldots e^{\delta_t \hat{L}_1/2}$$

Operator splitting (cont)

One splitting for the Maxwell-Boltzmann equations:

$$e^{\delta_t \hat{L}} \sim e^{\delta_t \hat{L}_{ub}/2} e^{\delta_t \hat{L}_{ue}/2} e^{\delta_t \hat{L}_r/2}$$

$$e^{\delta_t \hat{L}_B/2} e^{\delta_t \hat{L}_E} e^{\delta_t \hat{L}_B/2}$$

$$e^{\delta_t \hat{L}_r/2} e^{\delta_t \hat{L}_{ue}/2} e^{\delta_t \hat{L}_{ub}/2}$$

where

$$\hat{L}_E : \partial_t \vec{E} = \varepsilon^{-1} \nabla \times \mu^{-1} \vec{B} - \varepsilon^{-1} \vec{J} - \varepsilon^{-1} \sigma \vec{E}$$

$$\hat{L}_B : \partial_t \vec{B} = -\nabla \times \vec{E}$$

$$\hat{L}_r : d_t \vec{r}_{s,n} = c \gamma_{s,n}^{-1} \vec{u}_{s,n}$$

$$\hat{L}_{ue} : d_t \vec{u}_{s,n} = \tfrac{q_s}{m_s c} \vec{E}\Big|_{\vec{r}_{s,n}}$$

$$\hat{L}_{ub} : d_t \vec{u}_{s,n} = \vec{u}_{s,n} \times \tfrac{q_s}{m_s \gamma_{s,n}} \vec{B}\Big|_{\vec{r}_{s,n}}$$

Time discretization

Repeatedly applying this splitting and grouping particle and field updates separately yields VPIC's field advance:

$$e^{\delta_t \hat{L}_B/2} e^{\delta_t \hat{L}_E} e^{\delta_t \hat{L}_B/2}$$

and VPIC's particle advance:

$$e^{\delta_t \hat{L}_r/2} I_J e^{\delta_t \hat{L}_r/2} e^{\delta_t \hat{L}_{ue}/2} e^{\delta_t \hat{L}_{ub}} e^{\delta_t \hat{L}_{ue}/2}$$

(I_J means J for the field advance is computed but the state is unchanged)

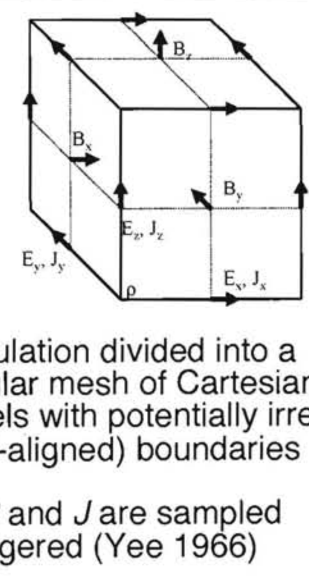
Thus, a mixture of explicit leapfrog, explicit exponentially differenced velocity Verlet and implicit Boris rotation is used to advance E, B and r from t to $t+\delta_t$ and u from $t-\delta_t/2$ to $t+\delta_t/2$

This underlying time discretization has robust theoretical properties; reversible, phase space volume conserving, ...

Los Alamos

NNSA

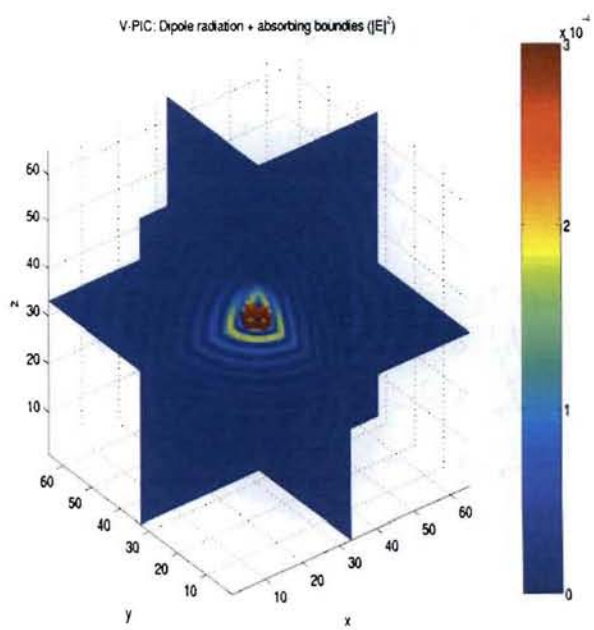
Space discretization



Simulation divided into a regular mesh of Cartesian voxels with potentially irregular (cell-aligned) boundaries

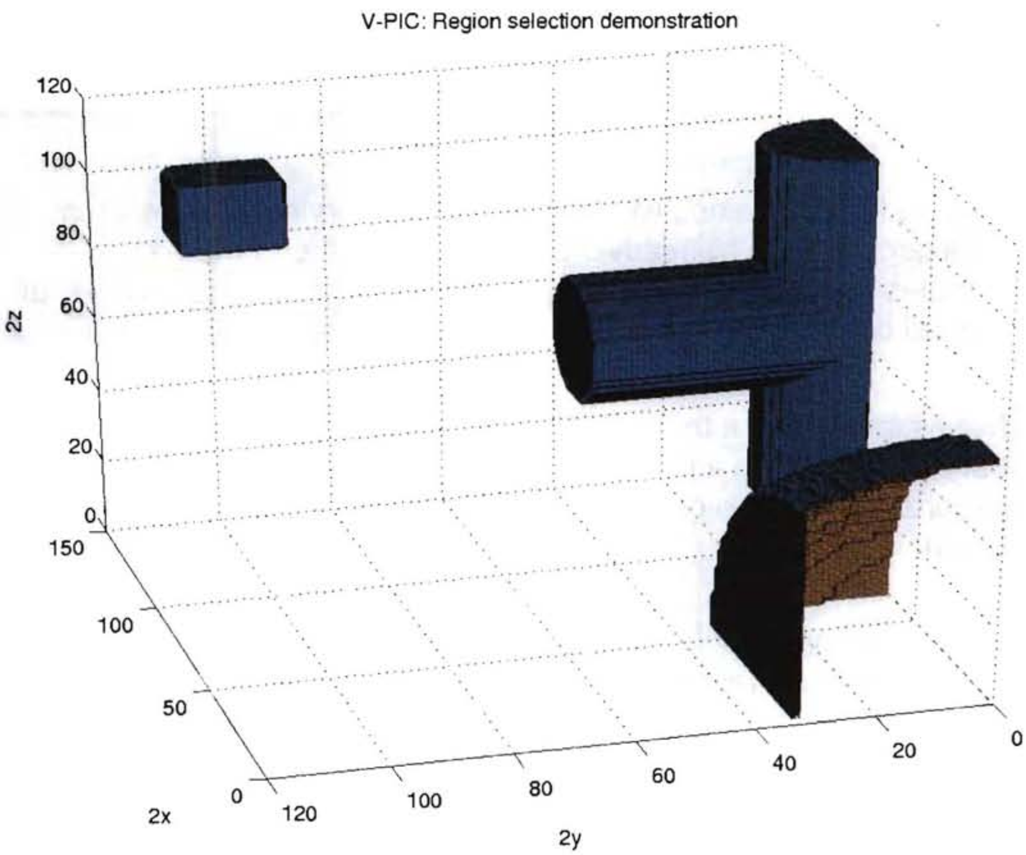
E, B and J are sampled staggered (Yee 1966)

Many boundary conditions supported (e.g., Higdon 1986)



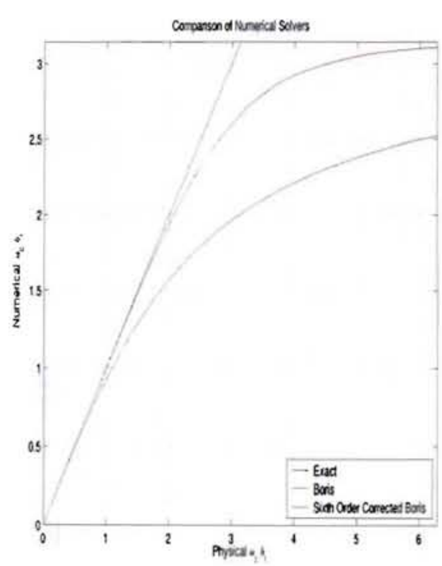
Los Alamos

NNSA



Particle advance

- Given particle fields, L_r, L_{ue} and L_{ub} can be applied exactly (in exact arithmetic)
- If L_{ub} applied exactly, high frequency cyclotron motion are aliased to lower frequencies
- 6[th] order L_{ub} approximation (reversible, energy conserving, phase-space volume conserving and unconditionally stable) used to prevent this (also used in Blahovec *et al* 2000)
- Nearly exact L_{ub} for low physical cyclotron frequency; asymptotes to Nyquist frequency otherwise and more efficient to compute



Los Alamos

ASC NNSA

Particle advance (cont)

Particle fields obtained with an "energy conserving" interpolation; for example, E_x is bilinearly interpolated from the four E_x edge samples and B_x is linearly interpolated from two B_x face samples of the cell containing a particle

Not as smooth as a trilinear "momentum conserving" interpolation but consistent with a finite element time domain formulation and it generalizes to more general meshing strategies (Eastwood *et al* 1995)

Easier to implement in simulations with non-trivial boundary conditions as no resampling of field components is required

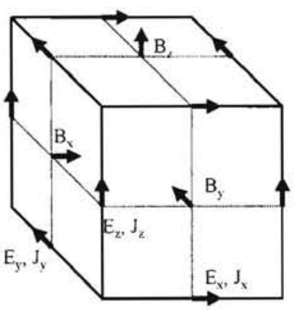
Particle advance summary

Interpolate Fields

$i_x, i_y, i_z, d_x, d_y, d_z$ = Cell and offset of $r^t_{s,n}$

$$E_x\big|^t_{s,n} = E_x\big|^t_{i_x+0.5,j_y,j_z}(1-d_y)(1-d_z) + E_x\big|^t_{i_x+0.5,j_y+1,j_z} d_y(1-d_z) + E_x\big|^t_{i_x+0.5,j_y,j_z+1}(1-d_y)d_z + E_x\big|^t_{i_x+0.5,j_y+1,j_z+1} d_y d_z$$

...



Accumulate Current (Inbounds shown)

$i_x, i_y, i_z, d_x, d_y, d_z$ = Cell and offset of $\vec{r}^{\,t+\delta_t/2}_{s,n}$

$$J_x\big|^{t+\delta_t/2}_{i_x+0.5,j_y,j_z} = J_x\big|^{t+\delta_t/2}_{i_x+0.5,j_y,j_z} + \frac{q_{s,n} v^{t+\delta_t/2}_{x,s,n}}{\delta_x \delta_y \delta_z}\left[(1-d_y)(1-d_z) + \frac{\delta_t^2 v^{t+\delta_t/2}_{y,s,n}}{12\delta_y\delta_z}\right]$$

...

Half Advance E

$$\vec{u}^{\,t^-}_{s,n} = \vec{u}^{\,t-\delta_t/2}_{s,n} + \frac{q_{s,n}\delta_t}{2m_{s,n}c}\vec{E}\big|^t_{s,n}$$

Boris rotation

$$\gamma^{t^-}_{s,n} = \sqrt{1 + \left|\vec{u}^{\,t^-}_{s,n}\right|^2}$$

α = Cyclotron freq correction

$$\vec{u}^{\,t^+}_{s,n} = \vec{u}^{\,t^-}_{s,n} + \left(\vec{u}^{\,t^+}_{s,n} + \vec{u}^{\,t^-}_{s,n}\right) \times \frac{q_{s,n}\alpha\delta_t}{2m_{s,n}\gamma^{t^-}_{s,n}}\vec{B}\big|^t_{s,n}$$

Half Advance E (2)

$$\vec{u}^{\,t+\delta_t/2}_{s,n} = \vec{u}^{\,t^+}_{s,n} + \frac{q_{s,n}\delta_t}{2m_{s,n}c}\vec{E}\big|^t_{s,n}$$

Advance Position

$$\gamma^{t+\delta_t/2}_{s,n} = \sqrt{1 + \left|\vec{u}^{\,t+\delta_t/2}_{s,n}\right|^2}$$

$$\vec{v}^{\,t+\delta_t/2}_{s,n} = c\vec{u}^{\,t+\delta_t/2}_{s,n} / \gamma^{t+\delta_t/2}_{s,n}$$

$$\vec{r}^{\,t+\delta_t}_{s,n} = \vec{r}^{\,t+\delta_t}_{s,n} + \delta_t \vec{v}^{\,t+\delta_t/2}_{s,n}$$

Field Advance

For diagonal tensor ε, μ and σ, given the curls, L_E and L_B can be applied exactly (in exact arithmetic)

Curls computed via 2nd order finite differencing

In finite precision, arithmetic error can cause Gauss' law violations to accumulate over time. To accommodate, VPIC periodically applies Marder passes (Marder 1987) tuned specifically to clean arithmetic error induced Gauss' law violations

While this method is local and inexpensive, because *J* is charge conserving, it suffices to use it infrequently to keep Gauss' law satisfied to near machine precision

Los Alamos

Field Advance (cont)

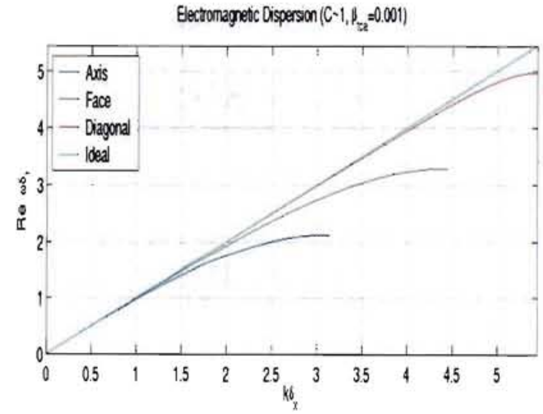
For short wavelengths, the discretized speed of light can deviate significantly from *c* and particles can generate non-physical Cherenkov radiation at these wavelengths

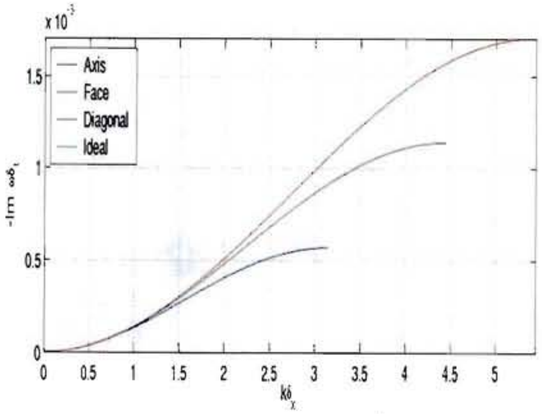
To reduce this noise, the background medium also has a tunable divergence free current response:

$$J_T = \tau \partial_t \left(J_T - \nabla \times \mu^{-1} B \right)$$

that damps this spurious radiation on a time scale τ

Same method used in Eastwood *et al* 1995





Los Alamos

Stability considerations

In vacuum, the field advance reduces to a FDTD method and the simulation must satisfy the Courant condition:

$$\left(\frac{c\delta_t}{\delta_x}\right)^2+\left(\frac{c\delta_t}{\delta_y}\right)^2+\left(\frac{c\delta_t}{\delta_z}\right)^2<1$$

Additionally, the particle advance usually requires:

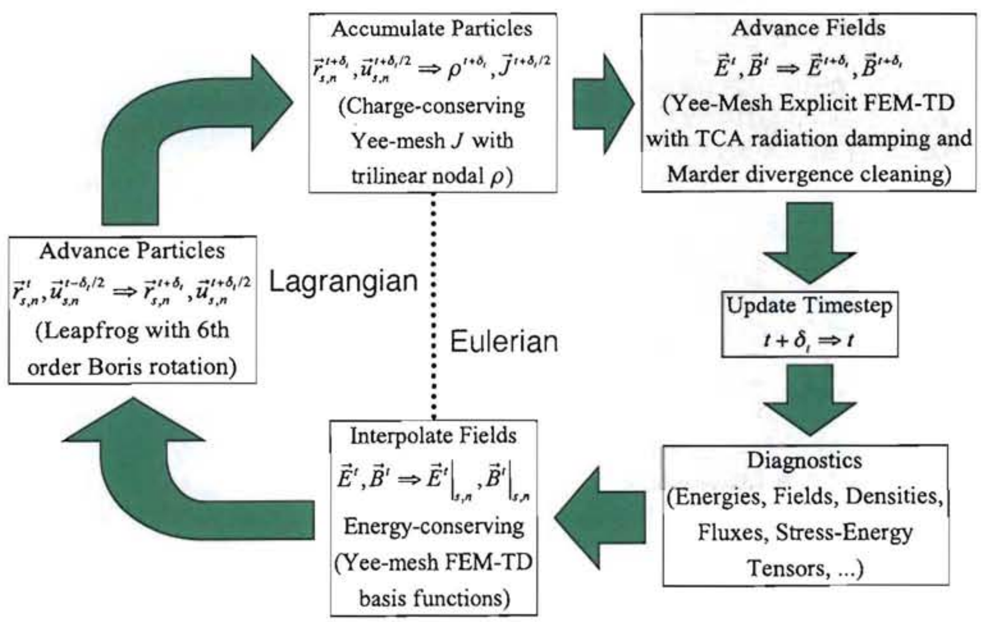
$$\omega_p\delta_t<2 \qquad \delta_{x,y,z}\approx\lambda_d$$

where ω_p is the plasma frequency and λ_d is the Debye length. Given particles cannot exceed c, satisfying the Courant condition and the Debye criterion typically is sufficient

Though simulations are stable for any physical cyclotron frequency, it is usually desirable to resolve it to keep dynamics accurate.

Sampling f_s typically requires between tens and thousands of particles per cell (depending on the simulation) to avoid non-physical computational particle collisional effects.

Methods summary



Structure of Arrays Versus Array of Structures Comparison

Memory hierarchies require a sorted AoS particle data layout for high performance.

Below calculations are for a minimal 2d2v electrostatic PIC simulation.

Pentium III 800/133 ATC
Dual channel RDRAM 800

FP Subsystem	M flop/s
3-cycle pipelined MAC	798

Memory Subsystem		M mop/s
Load	L1 cache	651
	L2 cache	427
	Memory	97.3
Store	L1 cache	664
	L2 cache	265
	Memory	**29.6**

Structure-of-Arrays (vectorized)

$$1.7\text{M pa/s} \approx \left(\frac{20 \text{ ldmm}}{97.3\text{M mop/s}} + \frac{10 \text{ stmm}}{29.6\text{M mop/s}} + \frac{49 \text{ flop}}{798\text{M flop/s}} \right)^{-1}$$

Array-of-Structures (thrashed)

$$2.0\text{M pa/s} \approx \left(\frac{16 \text{ ldmm}}{97.3\text{M mop/s}} + \frac{8 \text{ stmm}}{29.6\text{M mop/s}} + \frac{49 \text{ flop}}{798\text{M flop/s}} \right)^{-1}$$

Array-of-Structures (sorted)

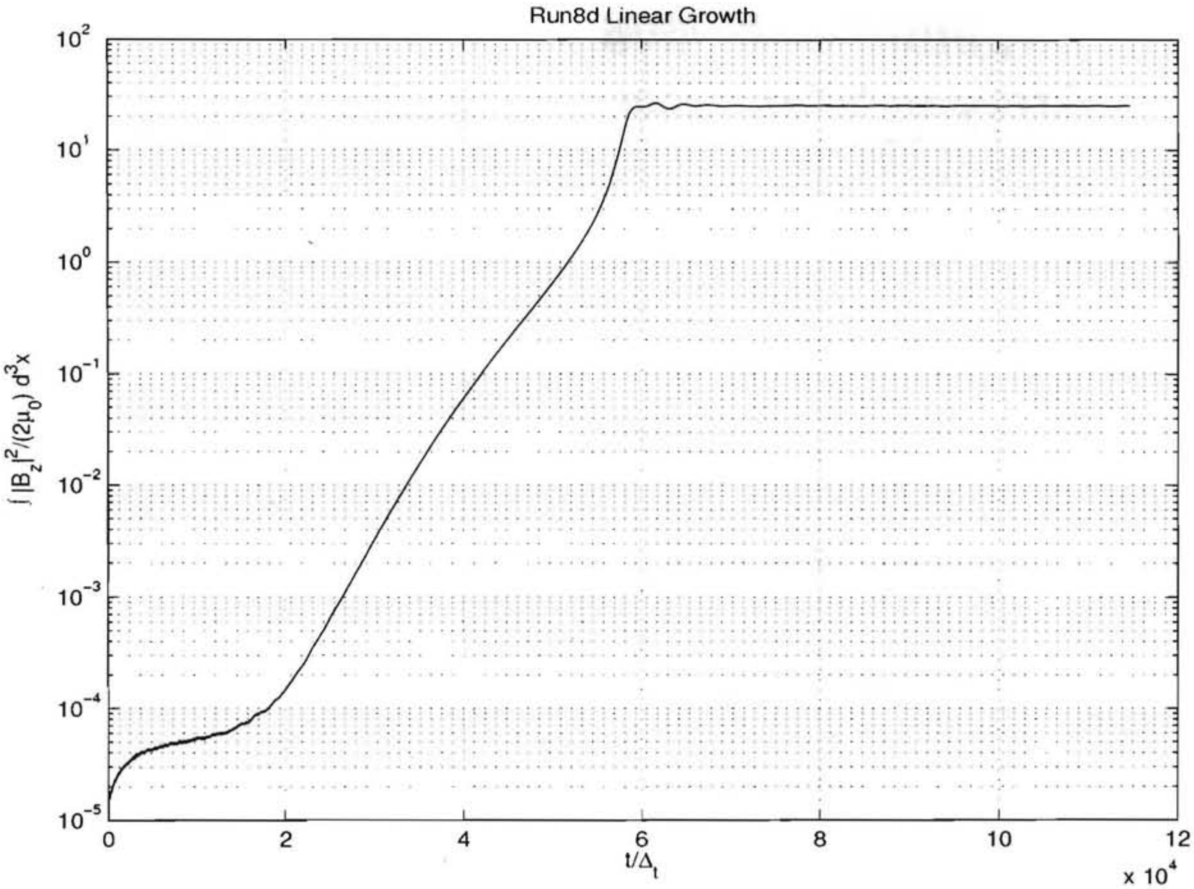
$$3.6\text{M pa/s} \approx \left(\frac{4 \text{ ldmm}}{97.3\text{M mop/s}} + \frac{12 \text{ ldl2}}{427\text{M mop/s}} + \frac{4 \text{ stmm}}{29.6\text{M mop/s}} + \frac{4 \text{ stl2}}{265\text{M mop/s}} + \frac{49 \text{ flop}}{798\text{M flop/s}} \right)^{-1}$$

Implicit Versus Explicit Cell Identification

- Conventional implicit particle-centric ("i_x=floor[x/δ_x]") is problematic.
  - Makes using anything but an axis-aligned uniform mesh hard.
  - Makes using single precision unsafe on large meshes as many bits of precision are used to resolve the mesh coordinates.
  - Many compilers implement float to integer operations very poorly (can reduce overall performance over ~50%).

- Implicit cell-centric (each cell tracks particles contained therein) can be cumbersome.
  - Memory management issues, esp. for non-uniform plasmas.

- Instead, particles explicitly store the index of the cell containing them.
  - It is the only viable strategy for non-uniform, curvilinear and unstructured arbitrary mesh partitions anyway.



Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use single precision aggressively

Pro: Half the data motion and wider SIMD available

Con: Requires great care for robust implementation

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Minimize passes through particles per step

Pro: Minimizes data streamed to/from DRAM

Con: Harder to modularize code; difficult to retrofit an existing code

Particle cache blocking a potential compromise

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use voxel + offset particle positions

Pro: Reduces position representation arithmetic error several orders of magnitude; essentially required for reliable single precision use; accelerates field interpolation, particle accumulation, particle sorting (especially on irregular meshes)

Con: Difficult to retrofit an existing conventional code

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Sort particles aggressively

Pro: Improved memory access temporal locality

Con: Determining the voxel (i.e., sort key) may be expensive on irregular meshes if using a conventional position representation

See SciDAC 09 paper for a description of VPIC's NUMA-friendly thread-parallel particle sorting algorithm

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use SIMD-friendly array-of-structures data layout

Pro: Improved memory access spatial locality, SIMD

Con: Difficult to retrofit an existing conventional code, ideal layout varies somewhat across methods and architectures (e.g., alignment restrictions).

FORTRAN is the albatross around the neck of HPC

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Do "cold" particle advance with 4-way vertical SIMD

Pro: Optimizes common case particle update

Con: Common case might not be common enough to make worthwhile in some simulations; wider SIMD architectures (e.g., GPUs are effectively ~16-way, Intel has 8-way and 16-way chips in development) may be more optimal under other strategies

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use an "interpolator"

Interpolate E,B from per-voxel interpolation coefficients computed before particle advance

Pro: Faster interpolation (especially on irregular meshes); improved memory access spatial locality

Con: Potentially prohibitive memory footprint for higher order methods; suboptimal when $< \sim 1$ particle per voxel

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use an "accumulator"

Accumulate particles to per-voxel accumulation coefficients and convert into ρ, J after particle advance

Pro: Faster accumulation (especially on irregular meshes); improved memory access spatial locality

Con: Potentially prohibitive memory footprint for higher order methods; suboptimal when < ~1 particle per voxel

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use an "exception" list

Pro: Fewer particle passes per step, reduced instruction cache pollution; improved code modularity by isolating slow application-specific code (e.g. custom boundary conditions) from fast particle advance

Con: Volumetric scaling exception costs may not be exceptional enough to warrant handling separately (most exception costs scale as boundary surface area)

Random Thoughts on Gyrokinetic PIC

Many VPIC optimizations apply

Use a charge-conserving accumulation

(or walk the mesh like one)

Pro: Improved physics accuracy, position-to-voxel calc potentially reduced from $O(\lg N_{local\ voxel})$ to $O(1)$ for irregular meshes, robust particle-boundary hit detection

Con: May be suboptimal if particles pass through many voxels per step on average or if an $O(1)$ hash based position-to-voxel calc already used

Random Thoughts on Gyrokinetic PIC

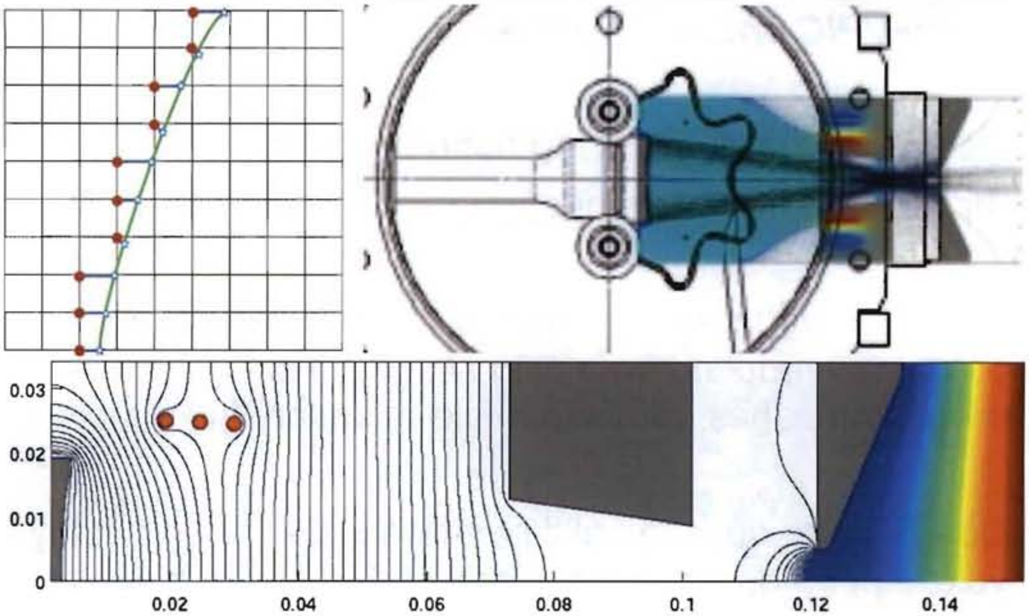
Other optimizations might be useful too

Use regular mesh with cut cells to accommodate irregular curved boundaries

Pro: Better understood numerics, easier to optimize than irregular meshes

Con: Might require additional research for use with gyrokinetics (e.g., Boltzmann electron cut-cell non-linear Poisson solvers)

Random Thoughts on Gyrokinetic PIC



Cut-cells in a H⁻ gun simulation (Chacon-Golcher and Bowers CiCP 2008)

Los Alamos

Random Thoughts on Gyrokinetic PIC

Other optimizations might be useful too

Use Hilbert space-filling curve voxel indexing

Pro: Improved temporal locality during particle advance (experimented with in VPIC on Roadrunner, minor gains for a regular mesh).

Con: Tricky to implement (especially for irregular and non-power-of-two meshes), position-to-voxel calculation possibly more expensive if using conventional position representation

Los Alamos

Random Thoughts on Gyrokinetic PIC

The Poisson equation is a bad idea in HPC

Information must propagate from each node to all others nodes every field advance

Non-local, elliptic-flavored field equations assume *c* (or the speed sound or ...) is effectively infinite on the grounds it is much faster than phenomena of interest

Requires expensive communications due to FFTs (regular meshes), reduce / broadcast communication trees (multipole and multigrid methods), ...

Random Thoughts on Gyrokinetic PIC

Use the Maxwell equations with a slow c instead

A slower *c* yields an increasingly scalable field advance and lenient Courant condition

The Poisson-Boltzmann and slow *c* Maxwell-Boltzmann systems can both model phenomena slower than the slow *c*. (These systems even have identical $Z(\beta)$ after integrating over the radiation field dof's.)

Could a "slow c" be used to make non-local gyrokinetic models more HPC friendly?